exemplary embodiments, a compute node **110** can include a data storage repository (not shown) for storing data including, but not limited to, EntityIDs, EventIDs, MinibatchIDs, PartitionIDs, sets of EventIDs, a Counters Table, a Streams Table, an Events Table, store minibatches and partitions. Data storage repository can be any programmable electronic device or computing system capable of receiving, storing, and sending files and data, and performing computer readable program instructions capable of communicating with driver **104**, one or more clients **108**, and stream computing program **102**, via network **106**. In an exemplary embodiment, a compute node **110** can be a commodity server capable of failing at any time.

[0024] In some exemplary embodiments, driver **104** generates a distributed key value store and distributes a copy of the distributed key value store to the one or more compute nodes **110**. In some exemplary embodiments, each compute node **110** in the cluster retains a cached copy of the distributed key value store. In other exemplary embodiments, driver **104** distributes the distributed key value store to one or more compute nodes **110**.

[0025] FIG. **2** is an example illustrating stream processing in a distributed computing environment **100**, according to an exemplary embodiment. FIG. **2** illustrates a distributed stream computing system that utilizes a distributed input data stream source, such as a receiver (not shown), one or more compute nodes **110**, and a distributed file system directory **206** to achieve exactly-once processing semantics.

[0026] In an exemplary embodiment, a distributed input data stream source generates data from a distributed messaging system (not shown) and provides the generated data as input for the system. The input data, such as a resilient distributed dataset (RDD), may be a series of minibatches, such as minibatch **201**. Each minibatch may be composed of one or more partitions, such as P**1** **202**, P**2** **203**, and PN **204**. A partition may be a subset of records in a batch that a compute node **110** may receive and process. A record may be a collection of fields that a compute node **110** may process (e.g. first name, last name, address, department, etc.). In an exemplary embodiment, a distributed input data stream source provides reliable input data, which is not corrupted in the event of a fault (e.g. a power failure) during computation at one or more compute nodes **110**.

[0027] In some exemplary embodiments, the distributed stream computing system routes the one or more partitions through one or more compute nodes **110** running a pure function. In an exemplary embodiment, the distributed stream computing system can route a partition to any one of the one or more compute nodes **110**. For example, the system may route P**2** **203** to Compute Node **2**. In another exemplary embodiment, the system can route two or more partitions to a compute node. For example, the system may route P**1** **202** and P**2** **203** to Compute Node **1**.

[0028] In some exemplary embodiments, a compute node **110**, running a pure function, processes one or more partitions. The distributed stream computing system stores the computed output of each partition in a Distributed File System file (i.e. a part file), such as DFS Part File **208**, in Distributed File System Directory **206**. The distributed stream computing system outputs the collection of all the part files.

[0029] When a compute nodes **110** faults (e.g. sustaining a power failure) during the processing period, the distributed stream computing system only partially processed the logical partition of input data for the faulted compute node **110**. The system achieves exactly-once processing semantics by re-processing the computation on that partition either on the same compute node or another compute node, resulting in over-writing of the part file containing the partially processed partition.

[0030] FIG. **3** is a functional block diagram depicting a storage database **300** of stream computing program **102** for distributed stream processing with non-idempotent output operations, according to an exemplary embodiment. In some exemplary embodiments, storage database **300** can be located locally on stream computing program **102**. In other exemplary embodiments, storage database **300** can be located remotely from driver **104** on one or more compute nodes **110**. In yet other exemplary embodiments, storage database **300** may be located on a secondary distributed stream processing system.

[0031] In an exemplary embodiment, stream computing program **102** receives input messages from one or more clients **108**, via network **106**. In other exemplary embodiments, stream computing program **102** receives input messages from a stream computing ingress system. Stream computing system **102** dispatches output messages as a response to the request from an environment. In another exemplary embodiment, stream computing program **102** may transport messages via a web-based protocol, such as REST, into a messaging bus. Messages may refer to various activities such as the amount of time a user spends on a website or the amount of money spent by a consumer on a retailer's website. Messages, entering or exiting a distributed computing system, are called events.

[0032] In some exemplary embodiments, each event contains two identifiers such as "EntityID" and "EventID." EntityID uniquely identifies the associated entity or user (e.g. customer, prospect, etc.) that relates to the event. In an exemplary embodiment, stream computing program **102** assigns an EntityID to the entity or user. In another exemplary embodiment, an external system can assign the EntityID. EventID is an event value assigned to the event. This event value is unique, across the one or more compute nodes **110**, to a given EntityID. In an exemplary embodiment, the event value can monotonically increase. In some exemplary embodiments, stream computing program **102** can assign the identifiers to the event at the ingress point. In an exemplary embodiment, the event include separate mini-batches, such as mini-batch **201**. In an exemplary embodiment, stream computing program **102** can logically model the mini-batches as a fault tolerant immutable, partitioned collection of elements. Mini-batch **201** includes one or more partitions, such as P**1** **202** and P**2** **203**. A partition, such as P**1**, includes multiple records. In an exemplary embodiment, stream computing program **102**, utilizing an ingress system, such as driver **104**, can dispatch each partition to a specific compute node **110**, such as compute node **1**, via network **106**. In some exemplary embodiments, stream computing program **102** can push the partitions to one or more compute nodes **110** using the logic of the partitioning scheme such as a key-range, hash function based partition, directory, or using the underlying distributed system.

[0033] In some exemplary embodiments, storage database **300** can be a key value store. The key value store can include three columns, table **302**, key **304**, and payload **306**. Table **302** includes three tables, counters table **308**, streams table **309**, and events table **310**. Counters table **308** includes